Instructions: Answer any three questions.
                         All questions carry equal marks.

                         2 hours
Duration
No. of Pages   7
Department(s)   Information Technology
Course Co-ordinator(s)

Requirements:
MCQ   Release to Library: Yes [   ]   No [   ]
Handout
Statistical/ Log Tables
Cambridge Tables
Graph Paper   Required
Log Graph Paper
Other Materials

1

Q1.

a) Consider the OpenGL program below, which defines camera parameters for perspective projection, positions the camera, and displays a GLUT teapot primitive at the centre of the root coordinate system. **Note that the final page of this exam paper summarises some useful OpenGL functions.**

```
#include <glut.h>

void display(void) {
   glMatrixMode (GL_PROJECTION);
   glLoadIdentity ();
   gluPerspective( 45.0, 1.0, 1.0, 50.0 );
   glMatrixMode(GL_MODELVIEW);
   glLoadIdentity();
   gluLookAt( 7.0, 7.0, 9.0,    0.0, 0.0, 0.0,    0.0, 1.0, 0.0);

   glClear (GL_COLOR_BUFFER_BIT);
   glutWireTeapot( 3.0 );
   glFlush ();
}

int main(int argc, char** argv)
{
   glutInit(&argc, argv);
   glutInitDisplayMode (GLUT_SINGLE | GLUT_RGB);
   glutCreateWindow ("Teapot!");
   glutDisplayFunc(display);
   glutMainLoop();
   return 0;
}
```

   i.   Extend this program so that it uses a Timer callback to animate the scene at a rate of 50 updates per second. The animation should involve the teapot spinning around its y ('up') axis at a rate of 1 full revolution per second.

[5 marks]

   ii. Explain why it is inefficient to have the camera setup code in the **display()** function.

[2 marks]

   iii. Modify the program so that it uses the matrix stack (**glPushMatrix();** and **glPopMatrix();**) in order to improve efficiency

[4 marks]

b) Consider the display of a realistic forest in an interactive 3D graphics environment: the key trade-off is frame-rate versus polygon count. In this context, discuss the use of textures, backface culling, levels of detail (LODs), mipmaps, bumpmaps and billboards in order to obtain a maximal frame-rate. Illustrate your answer with diagrams, and explain each technique carefully, indicating why it is useful.

[9 marks]

Q2.

a) Discuss how graphics hardware has changed (and improved) over the past decade. In your answer, refer to these terms: Graphics Processing Unit (GPU), on-board RAM, texture memory, data bus, DirectX/OpenGL   [5 marks]

b) Why are nested co-ordinate systems useful for 3D graphics/animation programming? In your answer, explain and provide code samples illustrating the use of nested co-ordinate systems in both OpenGL and X3D.

[5 marks]

c) Write X3D code to reproduce the 3D model depicted below, as closely as possible. The model consists of 8 identical extrusions with oval-shaped cross-sections, and a semi-opaque material is applied. Also provide sketches illustrating your cross-section co-ordinates and extrusion parameters. **Note that the final page of this exam paper lists the fields of some useful X3D nodes.**

[10 marks]
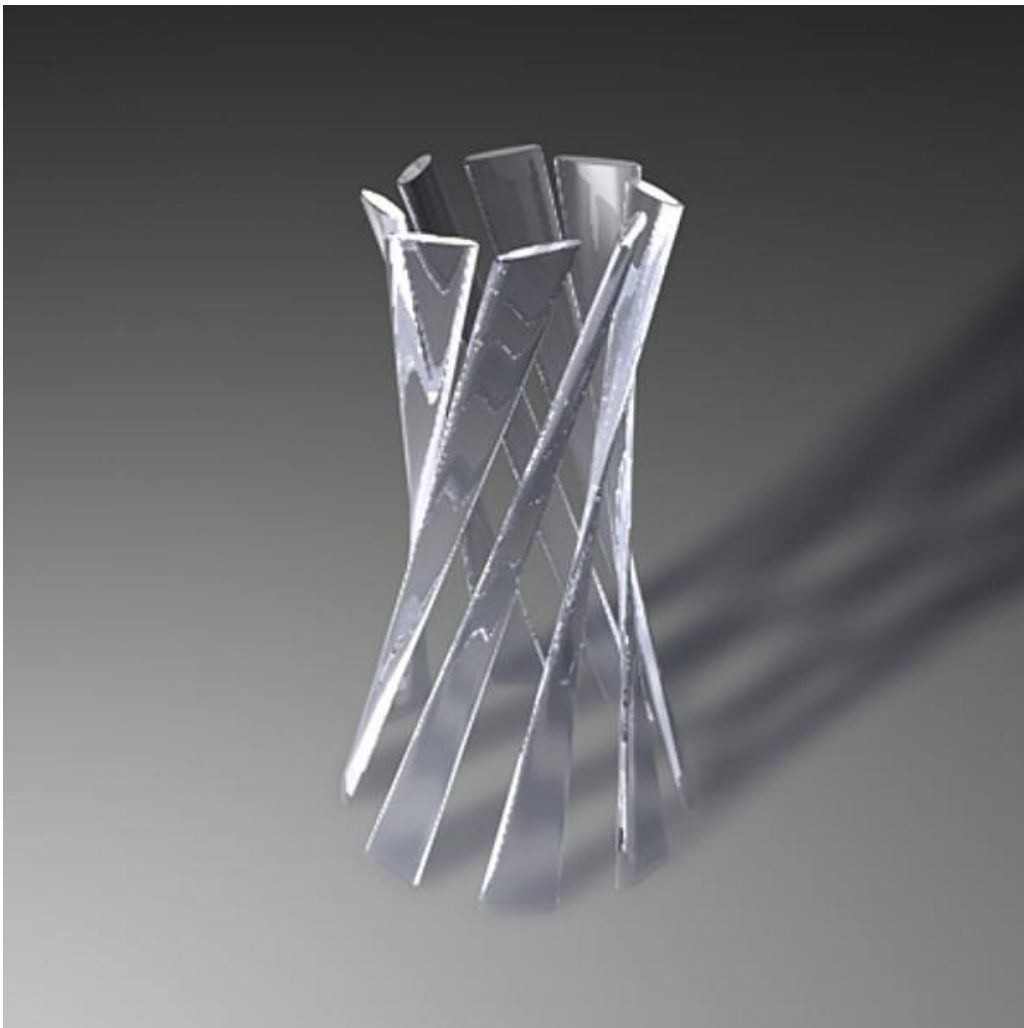
Q3.

a) In 3D computer graphics, what is a surface normal? Illustrate your answer with a diagram.   [3 marks]

b) Discuss, with the use of a diagram in each case, the following 3D graphics techniques. In each case, identify the importance of surface normals:

    i.   Flat (Lambert) Shading
    ii. Normal Interpolating (Phong) Shading
    iii. Radiosity
    iv. Binary Space Partitioning   [8 marks]

c) Explain the keyframe approach to animation in computer graphics, and explain its use in X3D, referring to the TimeSensor, Transform, OrientationInterpolator and PositionInterpolator nodes in your answer.
                                                                          [4 marks]

d) Rather than using hard-coded animation parameters, a more powerful approach to producing animations in X3D is to use JavaScript nodes to dynamically calculate key positions or orientations. Write X3D code for a JavaScript node which produces a smooth elliptical motion, and which is suitable for use with a TimeSensor node and a Transform node.
                                                                          [5 marks]

Q4.

a) Describe the *mathematical morphology* approach to image processing. Outline some typical circumstances in which this approach is useful. [4 marks]

b) The image below contains a large white rectangle and a number of patches of noise. Outline and defend a morphology-based algorithm for automatic isolation of the rectangle. Use sketches to indicate approximately how the image would appear following each step of your solution. [7 marks]
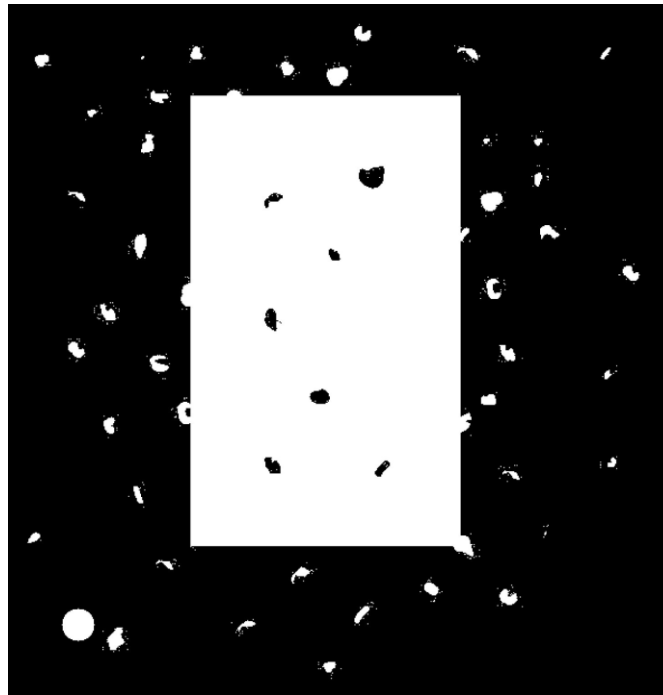
c) With regard to the use of image processing for the capturing of 3 Dimensional (3D) information, define the terms: (i) active depth sensing, (ii) passive depth sensing. [3 marks]

d) Outline, at a high level, a suitable algorithm for extracting 3D information from a stereo pair of images, such as the image pair illustrated below. [6 marks]
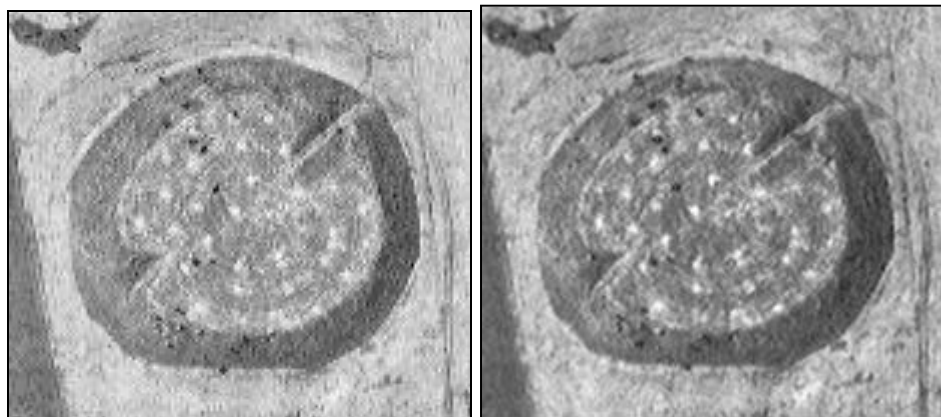
Q5.

a) Edge Detection is a basic image processing technique, which is often used as one of the earlier steps in image processing systems. Discuss the role of edge detection in a typical object extraction task. Why would the image resulting from edge detection be useful to the later image processing steps in your task? Be specific in your answer, choosing at least one image processing algorithm to use after edge detection, explaining why an edge-enhanced image is useful to this algorithm.

[6 marks]

b) Outline the following algorithms, and discuss their relative strengths and weaknesses: (i) convolution-based edge detection, (ii) compass edge detection, (iii) the 'Canny' algorithm. Provide diagrams in each case.
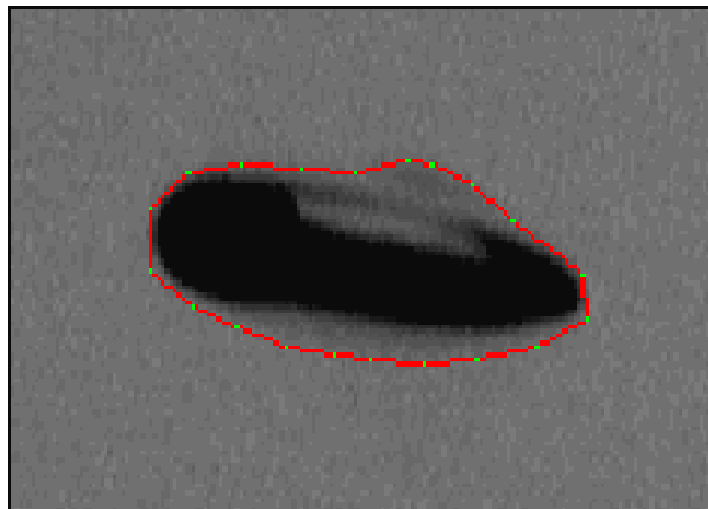
[6 marks]

c) Outline and discuss the image processing technique called 'active contours'.

[4 marks]

d) Present a suitable set of optimisation constraints (sometimes called energy factors) for accurately tracing the outline of a bubble such as the one shown below, using active contours.

[4 marks]

Some useful X3D nodes:

Node	Important Fields and Nested Nodes
Shape	Nested Nodes: Appearance, Geometry Nodes (Box, Sphere, Cone, Cylinder, Text, Extrusion, etc.)
Appearance	Nested Nodes: Material, ImageTexture
Material	Fields: diffuseColor, specularColor, emissiveColor, ambientIntensity, transparency, shininess
ImageTexture	Fields: url
Transform	Fields: translation, rotation, scale, center. Nested Nodes: Other Transforms, Shapes, Sensors
TimeSensor	Fields: enabled, startTime, stopTime, cycleInterval, loop
PositionInterpolator	Fields: key, keyValue
OrientationInterpolator	Fields: key, keyValue
Extrusion	Fields: crossSection, spine, scale, orientation, beginCap, endCap, creaseAngle
Box	Fields: size
Sphere	Fields: radius
Cylinder	Fields: radius, height, side, top, bottom
Cone	Fields: height, bottomRadius, side, bottom
PointLight	Fields: on, location, radius, intensity, ambientIntensity, color, attenuation
ROUTE	Fields: fromNode, fromField, toNode, toField

Some useful OpenGL functions:

Function	Comments
glBegin(type), glEnd()	Together these delimit a set of vertices. 'Type' defines how to interpret them (GL_POLYGON, GL_TRIANGLES, GL_TRIANGLE_STRIP, GL_TRIANGLE_FAN, GL_QUADS)
glVertex2f(x,y), glVertex3f(x,y,z)	Define a vertex
glColor3f(r,g,b)	Define the current colour state of OpenGL (red, green, blue)
glMatrixMode(matrix)	Apply subsequent transformations to the specified matrix (GL_MODELVIEW or GL_PROJECTION)
glPushMatrix()	Store the current state of the currently selected matrix (modelview or projection) on the matrix stack
glPopMatrix	Recover the currently selected matrix by popping the top values from the matrix stack
glLoadIdentity()	Remove any existing transforms on the current matrix
gluOrtho2D(left, right, bottom, top)	Define a 2D orthographic projection with specified world coordinate viewport
gluLookAt(eyex,eyey,eyez, atx,aty,atz, upx,upy,upz)	Position the camera (*eye* coords), target it (*at* coords) and define its 'up' vector (*up* coords)
gluPerspective(fov, aspect, near, far)	Define a 3D perspective projection with field-of-view (fov), aspect ratio, near and far clipping planes
glTranslatef(x,y,z)	Translate the coordinate system of the current matrix
glRotatef(a,x,y,z)	Rotate the coordinate system of the current matrix
glScalef(x,y,z)	Scale the coordinate system of the current matrix along each of its principal axes
glutDisplayFunc(myfunction)	Register your display (render) function with glut
glutTimerFunc(delay, myfunction, data)	Register a function with glut for delayed execution – delay is in milliseconds